USARIEM TECHNICAL NOTE TN03-5

AN EXERCISE MODEL TO STUDY PROGRESSIVE MUSCLE FATIGUE DURING CONSTANT WORK RATE EXERCISE ON A CYCLE ERGOMETER

Prepared by

Charles S. Fulco[1], Sc.D., Paolo Bonato[2], Ph.D.,
Don Gilmore[2], M.S., and Steven F. Lewis[3], Ph.D.

[1]THERMAL AND MOUNTAIN MEDICINE DIVISION
[2]NEUROMUSCULAR RESEARCH CENTER
[3]SARGENT COLLEGE OF ALLIED HEALTH PROFESSIONS

July 2003

[1]U.S. Army Research Institute of Environmental Medicine
Natick, MA 01760-5007

[2,3]Boston University
Boston, Massachusetts 02215

DISCLAIMERS

Approved for public release; distribution is unlimited.

The views, opinions and/or findings contained in this publication are those of the authors and should not be construed as an official Department of the Army position, policy or decision unless so designated by other documentation.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 2003	USARIEM Technical Note

4. TITLE AND SUBTITLE
An Exercise Model to Study Progressive Muscle Fatigue During Constant Work Rate Exercise on a Cycle Ergometer

5. FUNDING NUMBERS

6. AUTHOR(S)
C.S. Fulco, P. Bonato, D. Gilmore, S.F. Lewis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Thermal & Mountain Medicine Division
U.S. Army Research Institute of Environmental Medicine
Kansas Street
Natick, MA 01760-5007

8. PERFORMING ORGANIZATION REPORT NUMBER
TN03-5

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Same as #7 above.

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

To gain insight into mechanisms of muscle fatigue in common everyday activities such as bicycling and running, those muscles which limit performance should be studied by tracking the fatiguability of the same muscles during the activity. However, conventional ergometric testing modes such as stationary cycling or treadmill exercise do not readily lend themselves to quantitating the progressive increase in muscle fatigue, characterized by a loss of muscle power. It has therefore not been possible to link the inherent absolute power losses directly to other serially acquired physiological, biochemical, and psychological measurements during constant work rate exercise. Doing so would allow the gradually increasing muscle fatigue --- and its potential causes --- to be systematically and accurately assessed.

To overcome this limitation, hardware and software modifications were made to a commercially-available cycle ergometer and computer (equipped with an analog/digital board and LABVIEW© software), respectively. Described in detail are the modifications allowing quantitation of the progressive loss of power during sustained conventional cycle ergometer exercise. The modifications include rapid and frequent user-dictated switching between a constant work exercise mode to produce the progressive power loss and a maximal isokinetic pedal revolution mode that periodically assesses power loss during ongoing exercise. There also is virtually no delay or change in position between the exercise and test modes.

This exercise model should provide an effective framework to study changes in previously unquantifiable power loss during dynamic cycle exercise and will allow direct comparison to other temporally related measurements under a variety of environmental, nutritional, and pharmacological interventions. The model will likely provide unique insights into involved mechanisms that are associated with muscle fatigue during conventional cycle ergometer exercise.

14. SUBJECT TERMS
Muscle fatigue, muscle power, isokinetic, cycling, cycle ergometer

15. NUMBER OF PAGES
38

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

TABLE OF CONTENTS

LIST OF FIGURES

Figures ..... **Pages**

ACKNOWLEDGMENT

The authors would like to thank Robert Soares for providing much of the day to day technical "linkage" between the USARIEM and the NeuroMuscular Research Center of Boston University.

EXECUTIVE SUMMARY

To gain insight into mechanisms of muscle fatigue in common everyday activities such as bicycling and running, those muscles which limit performance should be studied by tracking the fatiguability of the same muscles during the activity. However, conventional ergometric testing modes such as stationary cycling or treadmill exercise do not readily lend themselves to quantitating the progressive increase in muscle fatigue, characterized by a loss of muscle power. It has therefore not been possible to link the inherent absolute power losses directly to other serially acquired physiological, biochemical, and psychological measurements during constant work rate exercise. Doing so would allow the gradually increasing muscle fatigue --- and its potential causes --- to be systematically and accurately assessed.

To overcome this limitation, hardware and software modifications were made to a commercially-available cycle ergometer and computer (equipped with an analog/digital board and LABVIEW© software), respectively. Described in detail are the modifications allowing quantitation of the progressive loss of power during sustained conventional cycle ergometer exercise. The modifications include rapid and frequent user-dictated switching between a constant work exercise mode to produce the progressive power loss and a maximal isokinetic pedal revolution mode that periodically assesses power loss during ongoing exercise. There also is virtually no delay or change in position between the exercise and test modes.

This exercise model should provide an effective framework to study changes in previously unquantifiable power loss during dynamic cycle exercise and will allow direct comparison to other temporally related measurements under a variety of environmental, nutritional, and pharmacological interventions. The model will likely provide unique insights into involved mechanisms that are associated with muscle fatigue during conventional cycle ergometer exercise.

INTRODUCTION

PHYSIOLOGICAL OVERVIEW

Various models have been used to study human muscle fatigue during dynamic exercise, but the mechanisms remain poorly defined (1). Because of this paucity of information of a very important issue, the USARIEM, Natick, MA in collaboration with Sargent College of Boston University, Boston MA developed and validated in recent years a dynamic exercise model that features integration of dynamic knee extension exercise isolated to the knee extensor muscles with serial measurement of maximal voluntary static contraction (MVC) force of the same muscles (2; 6). This approach permits linkage between gradually falling force-generating capacity of dynamically exercised muscle with specific physiological, biochemical, and psychological factors under a wide variety of dynamic exercise and environmental conditions. In addition, during a single bout of exercise, several measures of voluntary muscle function are obtained --- MVC force of rested muscle (i.e., strength), rate of decline in MVC force during dynamic exercise (i.e., fatigue), time to exhaustion (i.e., endurance), percentage of MVC force reached at exhaustion, and MVC force recovery rate after exhaustion (2). The ability to demonstrate multiple effects of different experimental interventions has provided numerous unique insights into primary determinants of muscle performance during dynamic knee extension exercise (3-6; 9).

Despite the important physiological insights afforded by the dynamic knee extension model, the use of only MVC force (i.e., a static contraction) as an index to quantitate performance changes during small isolated muscle dynamic contraction is somewhat limited because of an inability to similarly assess the performance of much larger muscle group synergies such as during conventional cycle exercise. Moreover, since peak power output is a product of not only MVC force but also speed of muscle contraction, the peak power loss that is associated with muscle fatigue during sustained cycle exercise is therefore likely due to a combined loss of both MVC force

and peak contraction speed. To our knowledge, progressive power loss during conventional constant work rate exercise has not previously been quantitated.

The objective of the current project was to adapt many of the concepts previously developed for the small isolated muscle model (i.e., dynamic knee extension model (2)) and extend them to a larger muscle group exercise model for experimental applications that require higher metabolic rates to assess various environmental, nutritional, and pharmacological interventions. To that end, software and hardware modifications were made to an electrically-braked cycle ergometer to rapidly and repeatedly switch between a constant work exercise mode that produces progressive muscle fatigue (i.e., power loss) and a maximal isokinetic pedal revolution mode that periodically assesses the power loss during on going exercise.

Presented here are detailed documentation of the software and hardware modifications, as well as technical drawings and schematic diagrams.

PROJECT DEVELOPMENT OVERVIEW

Hardware and software modifications were made under contract with the NeuroMuscular Research Center (NMRC) at Boston University, Boston, MA to an existing USARIEM electrically-braked cycle ergometer (CardiO2 Cycle Ergometer, MedGraphics, St. Paul, MN) and a laptop computer equipped with an analog/digital board and LABVIEW© software (National Instruments, Austin, TX). The ergometer and computer were temporarily moved to the NMRC for development and modification.

This project was conducted in two distinct phases. In phase one, the potential capability and suitability of the ergometer, and any additional hardware modifications that would be needed to meet the USARIEM performance specifications had to be assessed first. During this phase, it was determined that the ergometer, with the addition of hardware to monitor pedal crank angle and torque, would meet the proposed performance specifications. In phase two, the hardware modifications were implemented and tested. In addition, data acquisition and control software to monitor and program the operation of the ergometer was developed and tested.

Phase 1. Assessment of the Capability of the CardiO2 Cycle Ergometer

1. Obtain Documentation

All required documentation for the CardiO2 Cycle Ergometer, including the Operators Manual, Service Manual, and control software command list were obtained and read thoroughly.

2. Assess the Modes of Regulation

Each available mode of operation (constant work, constant speed, and constant torque) was tested using the "Stand Alone Mode" of operation. However, this mode of operation did not have the flexibility to quantitatively assess options such as time constant and torque, or speed thresholds. These actions were implemented as discussed below.

3. Exploit LABVIEW Modules for Serial Commands

The serial communications interface with the CardiO2 Cycle Ergometer was first tested using a terminal emulator via computer. The purpose of this test was to determine if bidirectional communications could be established with the ergometer using the existing set of MedGraphics control commands. Upon successful completion of these tests, LABVIEW© was used to send sequences of serial commands to configure and operate the ergometer.

4. Assess Control Delays and Mechanical Response of the Ergometer

LABVIEW© was used to assess delays occurring as a result of the communications link, microcontroller program delay, and the mechanical response of the CardiO2 Cycle Ergometer. The electromechanical response of the ergometer was considered instantaneous (<200 ms) especially where the system would be required to reduce speed or increase torque by applying a braking force to the flywheel. All control commands or responses at 9600 baud were initiated within 2 ms.

Changes from constant to isokinetic exercise modes were seamless, exhibiting no lag or abrupt transitions using the preprogrammed time constants. (It was determined that faster and slower transitions could be programmed using other time constants, if needed.)

5. Determine the Feasibility of Adding Sensors to Monitor Crank Angle and Torque

The strain gage signal conditioning circuitry was located on the IPC Automation preamplifier board. The voltage offset and scale factor for the torque output were measured. It was determined that the values would be compatible with the National Instruments A/D board with no further conditioning required. The board was modified with an additional D sub-miniature connector so that the crank torque output signals from the preamplifier board and the regulated power for angle potentiometer excitation were available.

A crank angle sensor based on a rotary potentiometer was designed to provide non-volatile indication of the crank position. The design uses the potentiometer in ratiometric mode, where the output voltage is directly proportional to the angle of the crankshaft. A grooved belt transmits the rotation of a pulley mounted on the crankshaft to a similar pulley mounted on the shaft of the potentiometer. The potentiometer with a tension adjusting bracket is mounted internally on the ergometer frame. (One potential drawback to this design is a gap of several degrees as the crankshaft makes the transition from 360 degrees back to 0 degrees to complete one full pedal revolution. This gap is very small and can be adjusted by software. An alternative angle sensing device which has no gap in the transition could be up substituted for the existing potentiometer if this level of precision is require for future research endeavors.) Appendix II contains schematics and mechanical drawings used to fabricate the required hardware modifications.

Phase 2. Software Development for Programming Test Protocols

Software Logic Overview

The NMRC developed software tools that can be utilized and modified in the LABVIEW© environment in order to: 1) generate sequences of commands to program test protocols implemented using CardiO2 Cycle, 2) acquire up to 16 channels of generic analog inputs, and 3) send commands via the serial port to CardiO2 Cycle set with appropriate timing. These functions were implemented in separate Virtual Instruments (VIs), i.e., LABVIEW© modules dedicated to accomplish each task.

The first VI *(ScriptGen VI)* was developed to generate sequences of commands that are later loaded and sent via the serial port to the ergometer by another VI *(BikeController VI)*. Two modules were implemented to acquire data. The first of these modules *(ACQmoduleCH2 VI)* was designed to acquire data from two channels. This is necessary to display and send data to the hard-drive that are collected from the analog outputs provided by the ergometer after the hardware modifications. This allows continuous acquisition of crank position and torque. The second of these acquisition modules *(ACQmoduleCH16 VI)* can record 16 generic analog inputs.

The following provides a description of the VIs developed at the NMRC for the USARIEM. Included will be a presentation of the features available through the front panel of each VI as well as a summary of the characteristics of the VI diagrams for possible software modifications.

ScriptGen VI to Generate Sequences of Commands

ScriptGen VI is the first executed in order to program and implement a test protocol using the CardiO2 Cycle. FIGURE 1 shows the front panel of this VI. The use of the features provided by *ScriptGen VI* requires that the *Operate Value* of the Tools *Palette* be utilized. FIGURE 1 shows how the *Tools Palette* pops up on the computer screen.

FIGURE 1: Front panel of the VI (ScriptGen) designed to generate a sequence of commands that are later executed by CardiO2 Cycle in order to implement test protocols that switch among modes of exercise.

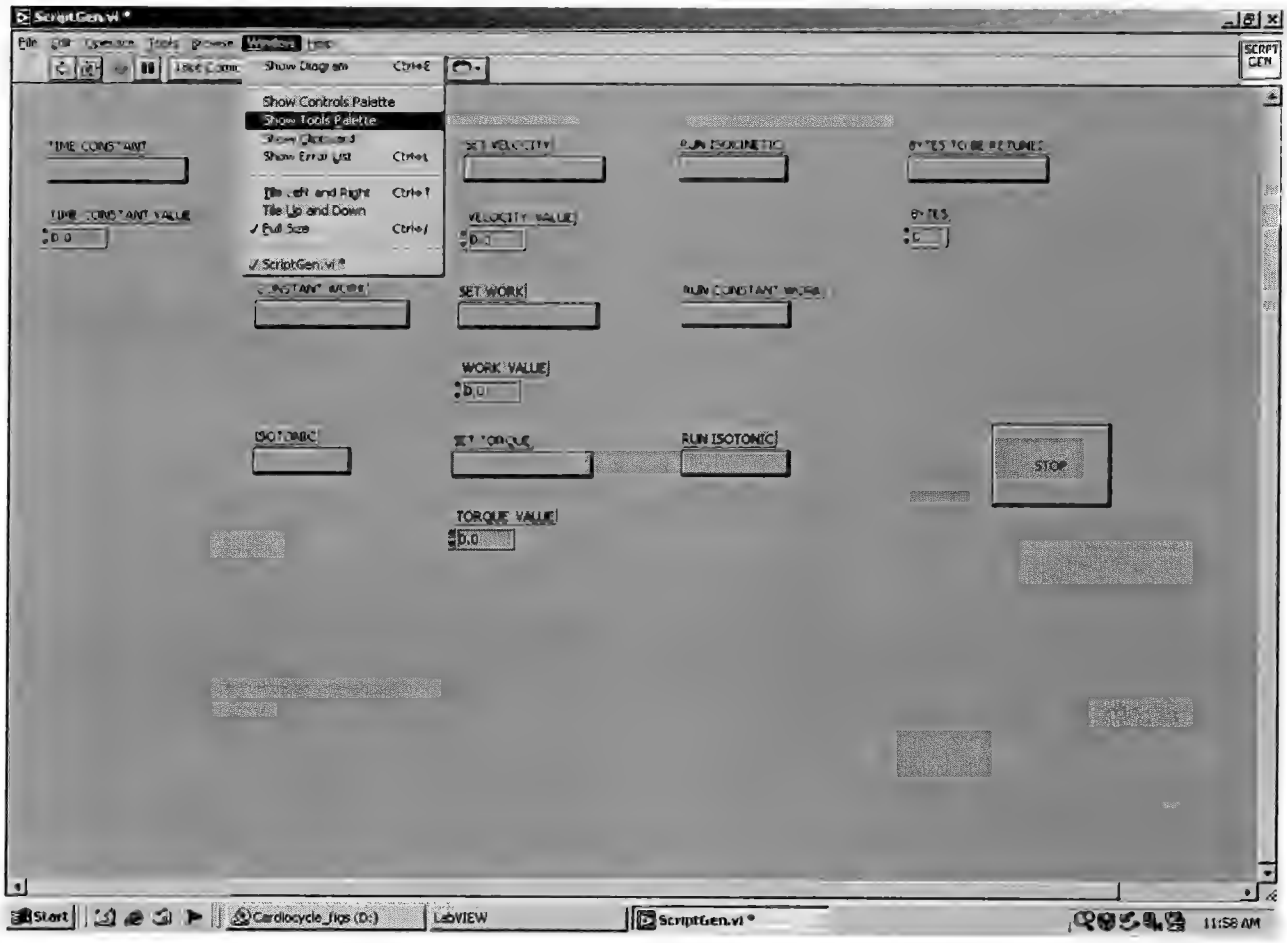
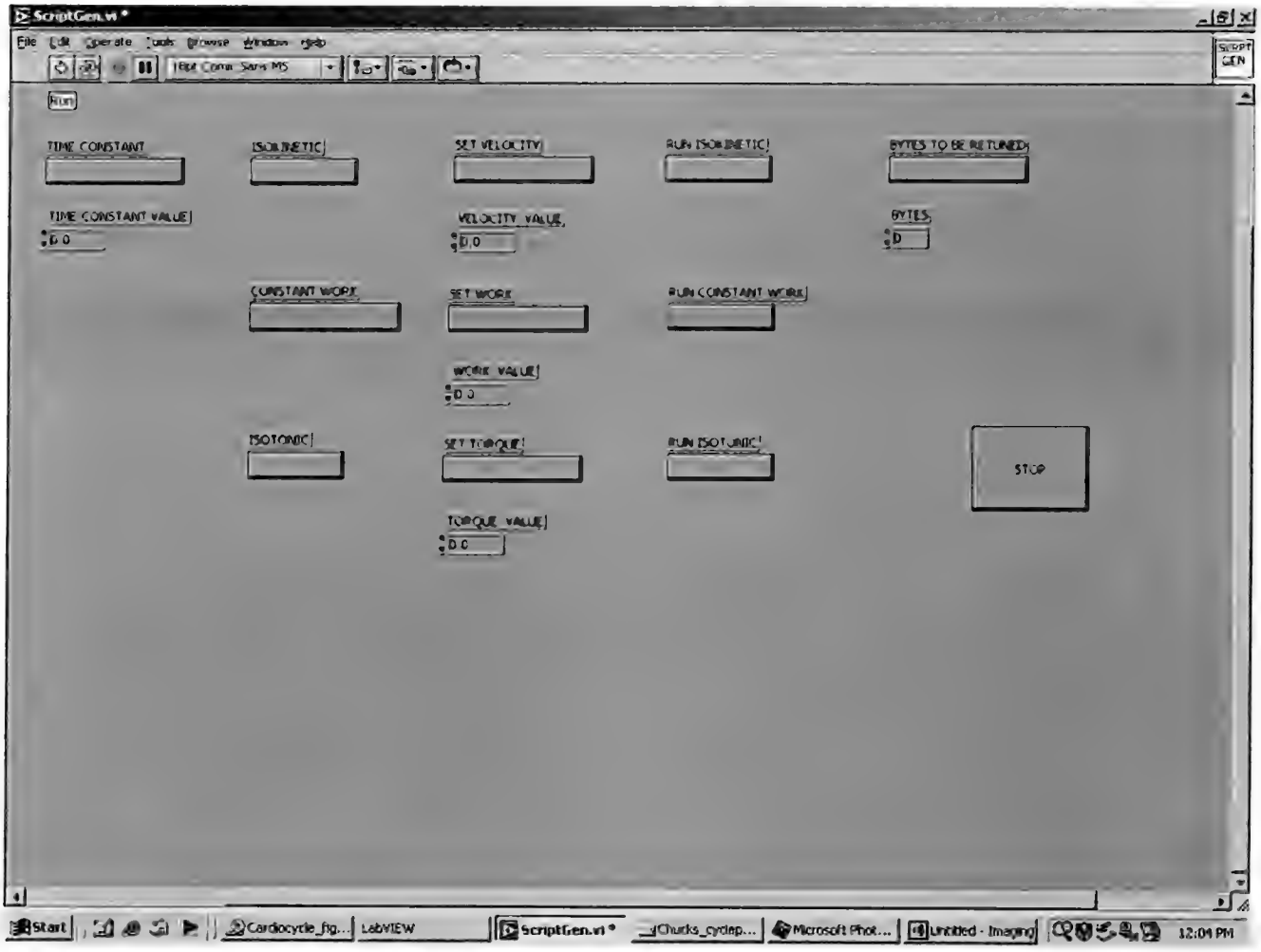


FIGURE 2 indicates the choice that has to be made in order to select the *Operate Value* function. The tool replaces the cursor after the selection is made. Using the *Operate Value* sets the digital controls to the desired numeric value. The digital control values can be modified by clicking on the arrows (up and down) on the left side of each number or by modifying the field by clicking on the number and inserting the desired numeric values utilizing the keyboard.

FIGURE 2: Front panel of ScriptGen with indication of the tool that the user has to choose in order to change the content of the digital controls. Once the digital controls are set, the corresponding push buttons may be used to set the selected function to the value shown by the digital control.

FIGURE 3: ScriptGen may be run by selecting the RUN button of the toolbar



The program may be started using the *RUN* buttons (FIGURE 3). After the execution of *ScriptGen VI* is started, the pushbuttons allow the choice of the following groups of functions: 1) set the time constant of the execution; 2) set the modality of the exercise; 3) set the parameters that regulate the chosen modality of exercise; 4) run the exercise; 5) set the number of bytes that constitute the response of CardiO2 Cycle to the last command sent via the serial port; and 6) stop the generation of the script file containing the sequence of commands to be sent to CardiO2 Cycle. The pushbuttons are set as "latch when pressed" which means that each button returns to the *unselect* position after the corresponding action is performed. Therefore each function is written in the script file once every time the corresponding pushbutton is selected.

Also, a *WAIT* module in the VI diagram guarantees that the pushbuttons appear as selected for a time interval long enough to allow the user to realize that the function has been written in the script file.

The time constant has to be set according to either the time that is necessary to allow the CardiO2 Cycle to respond to the command sent via the serial port or the time that a certain type of exercise needs to be executed. When setting the mode of exercise or its corresponding parameter, it is advisable to allow 100 ms to obtain a response from CardiO2 Cycle and about 1-2 sec for the response to display and provide a means to check for possible errors in the script file or anomalous responses of the CardiO2 Cycle. When the pushbuttons to run an exercise are selected, the time constant is the time interval during which the exercise is executed.

For instance, if an isokinetic exercise at 60 rpm for 30 sec is to be run, the sequence of selections are:

1) modify *TIME CONSTANT VALUE* to 100 ms and select the pushbutton *TIME CONSTANT,*
2) select the pushbutton *ISOKINETIC* and the *BYTES TO BE RETUNED* (as discussed below);
3) modify *SET VELOCITY* to 60;
4) select the pushbutton *TIME CONSTANT,*
5) select the pushbutton *ISOKINETIC* and the *BYTES TO BE RETURNED;*
6) modify *TIME CONSTANT VALUE* to 30 and select the pushbutton *TIME CONSTANT;*
7) select the pushbutton *RUN ISOKINETIC* and the *BYTES TO BE RETUNED;* and
8) select the pushbutton *STOP.*

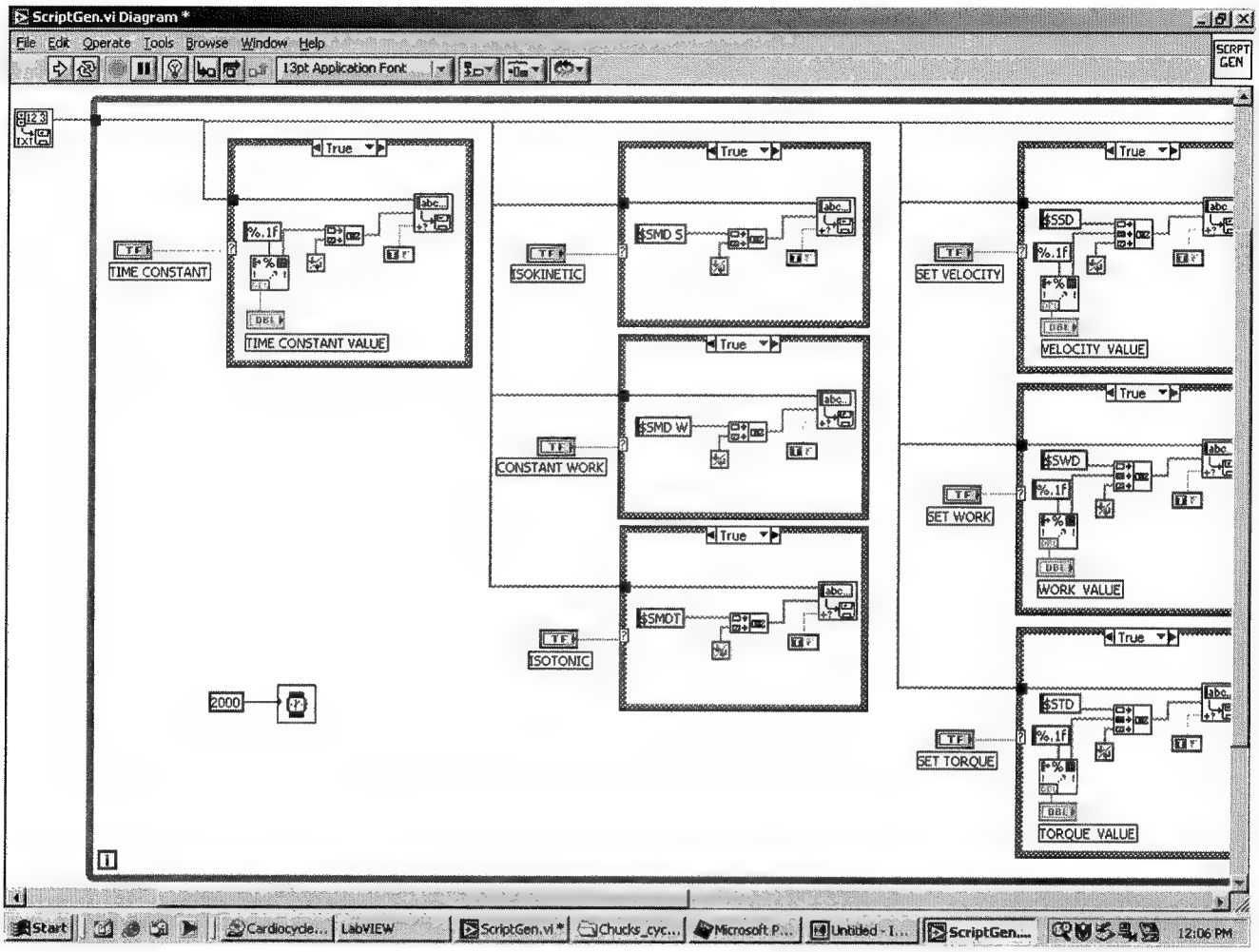
How these functions are implemented is shown by the VI diagram illustrated in FIGURES 4 and 5. A *WRITE TO SPREADSHEET FILE* VI (first module on the left side of the VI diagram) allows the user to select the output file name (a dialog box pops up when the execution of the program is started) with the possibility of overwriting previous script files erroneously generated or not useful anymore. A series of *IF* structures are then linked to each pushbutton: the *TRUE* case is shown while the *FALSE* case is not since the *FALSE* case was programmed for all *IF* structures to be empty. When a numeric value is associated with the pushbutton, the corresponding *IF*

structure contains a *FORMAT INTO STRING* module, a *CONCATENATE STRINGS* module, and a *WRITE CHARACTERS TO FILE* module. When a pushbutton corresponds to a function that does not require a numeric value, a string is written to the script file going only through the *CONCATENATE STRINGS* module. One input of this module allows adding an *END OF LINE* character so that each command is written in a different line of the output script file.

A *MILLISECONDS TO WAIT* module (see FIGURE 4) slows down the execution of this VI in order to make it possible to the user to realize that a pushbutton has been selected and the corresponding function has been executed. In fact, the *LATCH WHEN PRESSED* modality is otherwise executed so quickly that the pushbutton returns to the *unselect* position in less than a second, which may be too short a time to allow the user to follow the execution of the program. The STOP pushbutton is linked to the input of the module that controls the WHILE loop through a *NOT* module and therefore the execution is stopped when this pushbutton is selected.

FIGURE 4 shows the implementation of the first two sets of functions, i.e., those to set the time constant and those related to setting the mode of the exercise to be programmed. FIGURE 5 illustrates the modules related to the other three functions implemented by this VI to generate the script file containing the commands for CardiO2 Cycle, i.e., the modules to set the parameters of each exercise mode, those to run the exercise mode for a certain time interval, and the module to set the bytes to be returned by CardiO2 Cycle as responses to a command.

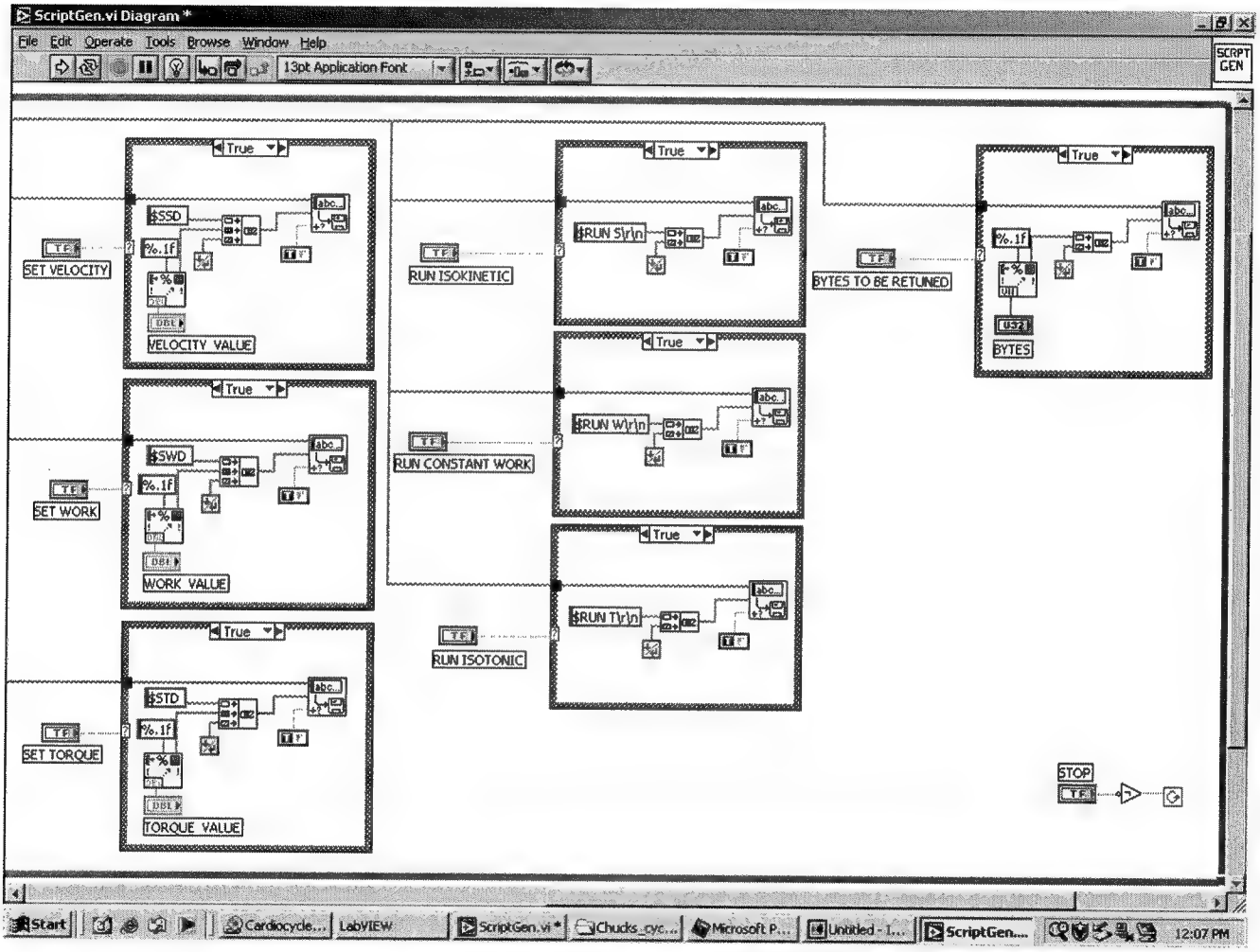
FIGURE 4: Diagram of ScriptGen. The main portion of the program is executed within a WHILE LOOP that continues until the user selects the STOP pushbutton. A series of IF structures implements the selection of the time constant and the mode of exercise.



Once generated, the script file can be viewed and, if necessary, modified using any editor suitable for manipulating text files (in ASCII format). Each line of the script file corresponds to either a time constant value, or a command to be executed by CardiO2 Cycle, or a number of bytes that constitute the length of the response of CardiO2 Cycle to the command sent via the serial port. When a new protocol is generated, it is advisable to run the protocol once before performing an actual test in order to verify that all the settings were properly programmed. The response of CardiO2 Cycle, as displayed by the window on the front panel of the *BikeController* VI, may be used to verify the correct functioning of CardiO2 Cycle under the programmed

protocol. In fact, the response to each command sent via the serial port provides information about the settings of CardiO2 Cycle.

FIGURE 5: Diagram of ScriptGen. Three sets of functions are illustrated. They provide the following functions: set the parameters of the exercise, run the CardiO2 Cycle in the desired mode of exercise, and set the number of bytes to be returned by CardiO2 Cycle in response to a given command.



In case problems arise, it is recommended that the commands be executed one at a time and that each response be checked. This is possible by opening the serial port modules as described and illustrated in the *BikeController VI* section.

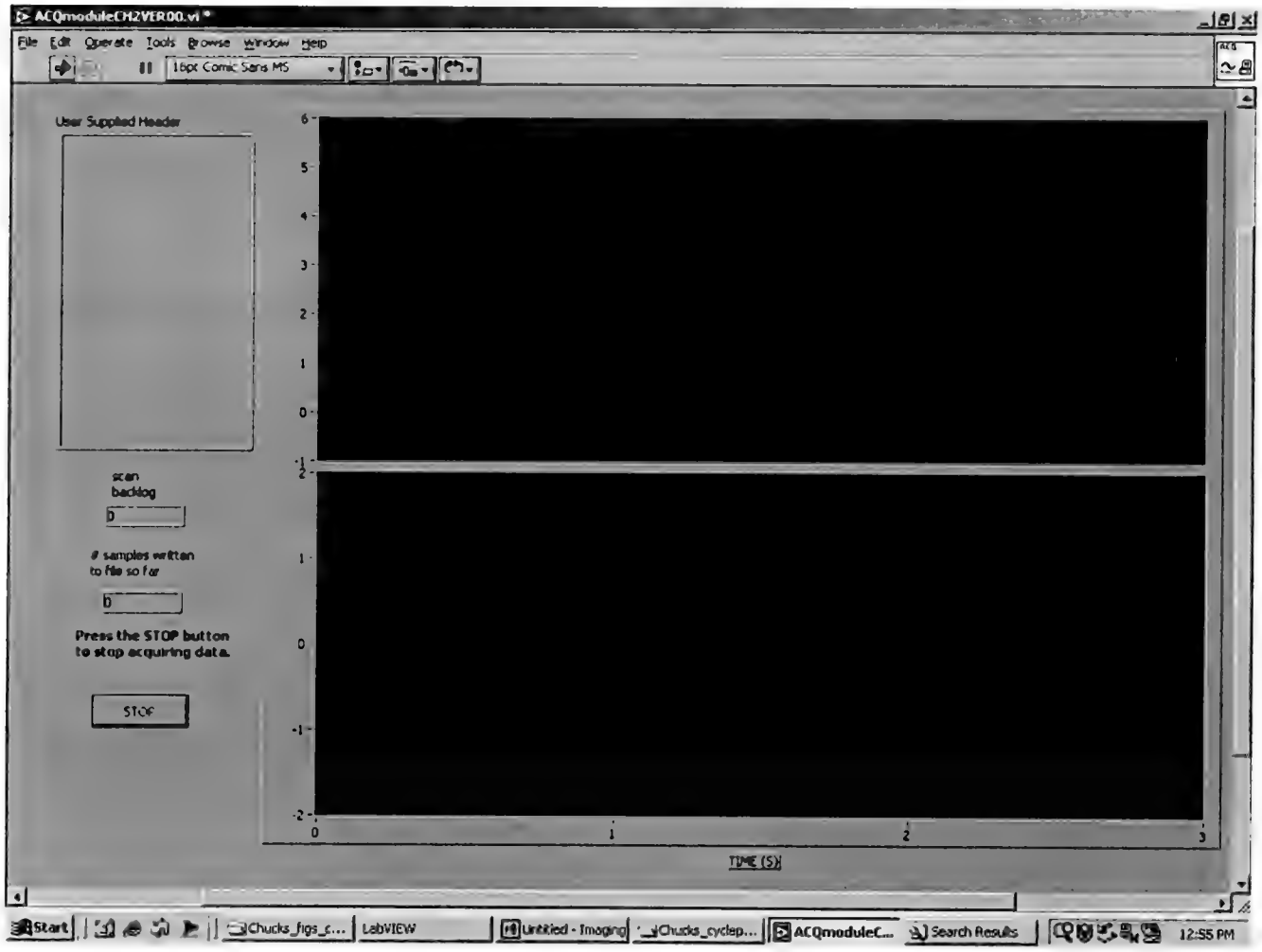
Finally, the number of bytes to be returned by CardiO2 Cycle needs to be set according to a summary table provided with the VIs (see text file CommandSet.doc included in the software). A general rule is that the echo provided by CardiO2 Cycle contains almost always the same number of bytes that constitute the command sent to

the device. However, exceptions to this general rule need to be checked in the documentation file enclosed in the folder containing the VIs.

ACQmoduleCH2 and ACQmoduleCH16: Two VIs to Record Data While Sending Commands to CardiO2 Cycle

Two VIs were written to acquire data from analog channels while sending commands to CardiO2 Cycle via the serial port using *BikeController*. The tests were carried out for protocols including isokinetic and constant work modes of exercise performed at 60 rpm, the work load varying from 50 to 200 watts, the duration of the isokinetic portions of the exercise varying from 3 to 5 sec, the duration of the constant work portions of the protocol from 10 to 60 sec, and the progressive increases and decreases of the work load set at a rate of 25 watts every 30 sec and 10 watts every 10 sec. In all these conditions, we were able to continuously display and stream to the hard-drive samples up to 16 channels at 1000 Hz for a duration of up to 5 min. FIGURE 6 illustrates the front panel of *ACQmoduleCH2*. This VI was specifically designed to display and record data collected from CardiO2 Cycle to monitor crank position and torque.

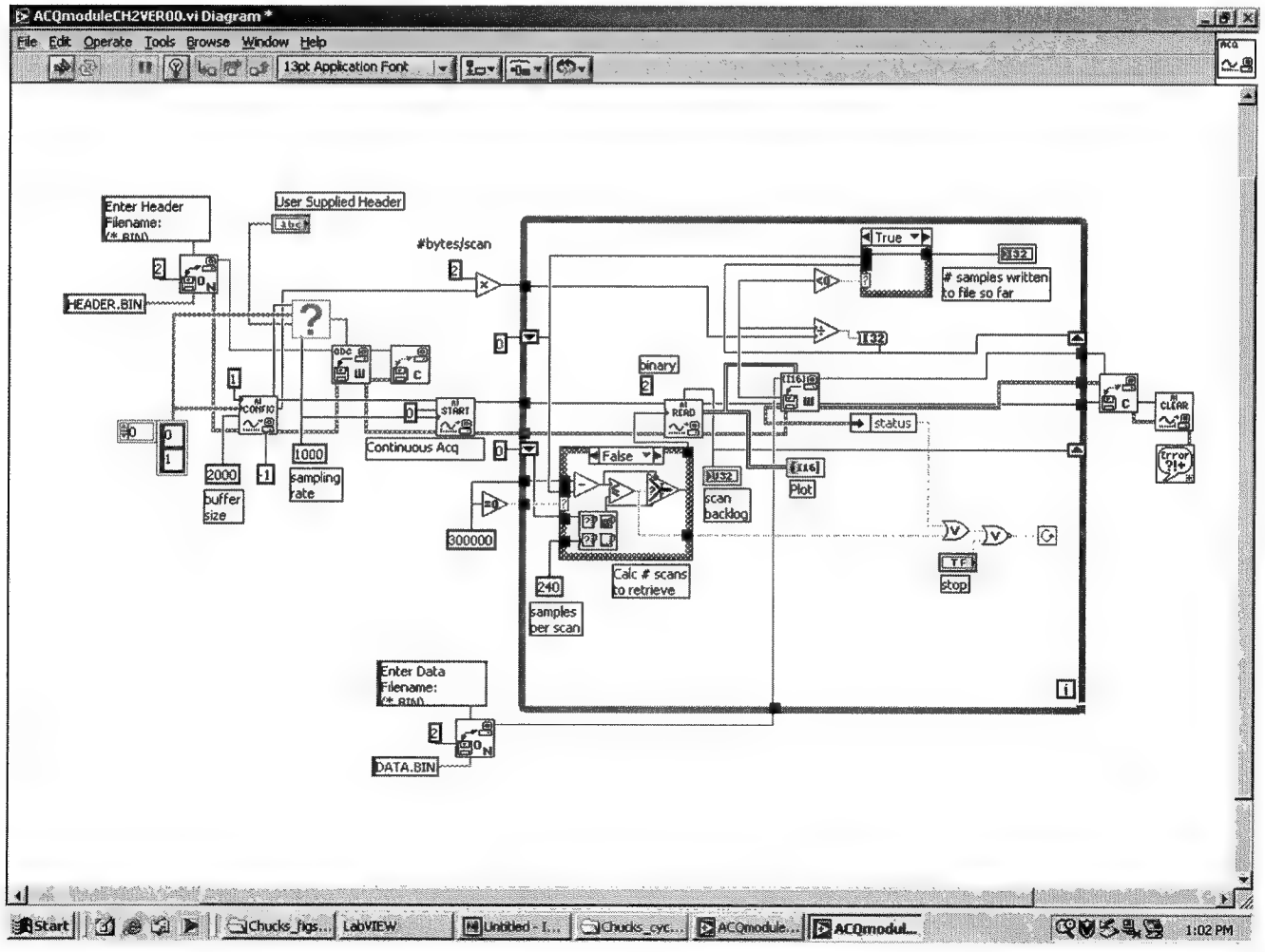
FIGURE 6: Front panel ofACQmoduleCH2. This VI allows the user to display and send to a binary file samples recorded from two analog channels. These were made available through hardware modifications of the CardiO2 Cycle to allow continuous monitoring of crank position and torque.



The VIs (for 2 and 16 channels) may be run using the *RUN* button provided by LABVIEW toolbar. Once started, the program may be stopped by using the *STOP* button or will be executed till the *WHILE LOOP* that controls the flow of the program (see FIGURE 7) is timed out. A header field is provided to input comments related to the settings of the experiment. This header is associated with the data file, but it is stored in a file separate from that containing the data samples. This characteristic of the design facilitates reading the data file for later analyses from a software package

other than LABVIEW©. An example is provided by means of a program written by the NMRC. This software allows importing and displaying data using MATLAB© *(The Math Works Co)* and is provided together with the LABVIEW© VIs. Finally, two digital indicators are provided to show possible accumulation of samples in the buffer utilized to transfer the data from the PCMCIA card to the hard-drive *(scan backlog)* and to indicate the number of samples already written in the data file at each point in time during the data collection.

FIGURE 7: Diagram of ACQmoduleCH2. The program is organized in two sections: the first one (on the left side) is devoted to opening and writing the header file, opening the data file, and initializing the PCMCIA card for data collection. The other section is contained in a WHILE LOOP that allows data displaying and data streaming to the hard-drive.



When the VIs implemented to collect 2 or 16 channels are executed, the user is prompted via dialog box to choose file name and the location of the header and data file. Then the program displays the data by means of waveform chart plots on the right side of the front panel. The VI diagram for the module designed to collect 2 channels is reported in FIGURE 7. The diagram of the module to collect 16 channels uses an almost identical structure. The diagram of this VI is divided into two sections: the first one relates to the initialization procedures, while the second section allows streaming the data to the hard-drive and its on-line display.

In the first section of the VI diagram, two *OPEN\CREATE\REPLACE FILE* modules cause two dialog boxes to appear on the screen when the VI is executed and the user is required to choose the filename and location of the header and data file. A *WRITE FILE* module linked to the *OPEN\CREATE\REPLACE FILE* module of the header file saves the content of the *STRING CONTROL* box in the front panel that is utilized to input notes related to the experiment. A *CLOSE FILE* module closes the header file before initiating the data acquisition. Two modules of *AI CONFIG* and *AI START* allow configuring the PCMCIA card and initiating the data collection. The configuration includes the allocation of a buffer for implementing buffered analog input operations. The inputs set in this VI relate to the device number (1 is the device number utilized using the PCMCIA card and computer), the channels (set by an array of strings) utilized for the data collection, the buffer size in number of scans (2000), an input error which is used to check for possible problems that occurred while opening the header file, and the interchannel delay which is set to -1 indicating that *AI CONFIG* uses the channel clock rate selected by LABVIEW© according to optimum criteria based on the hardware characteristics. For the *AI START* module the number of scans to acquire is set to 0, which indicates to LABVIEW© that the module sets the acquisition to continue indefinitely, i.e., until LABVIEW© stops the acquisition. The scan rate is set to 1000 scans per sec. The task ID and error input are derived from other modules whose execution logically precedes the execution of the *AI START* module.

The second section of the VI diagram is a *WHILE LOOP* that is executed until either the *STOP* pushbutton is pressed or the number of acquired samples exceeds 300,000. Two modules, *AI READ* and *WRITE FILE,* handle the data acquisition and

streaming to the hard-drive. The data are manipulated as integers represented by 16 bits and passed to the waveform chart as integers. This implies that the representation in volts is obtained by scaling the data directly using the plot functions, i.e., using the formatting options of the waveform chart. In case the user wants to modify the data representation, they have to right-click on the plot, select *Y SCALE* and *FORMATTING,* and when prompted, the *Y SCALE FORMATTING* dialog box, then modify the *SCALING FACTORS* options by inserting in the *DY* field the resolution of the AD converter (the current setting is for a range of ± 10 V and a 16 bit AD converter). The *AI READ* module outputs provide an important piece of information *(scan backlog)* that is displayed on the front panel to indicate a possible problem that might occur during a data collection. *Scan backlog* is the number of samples acquired minus the number of samples read. If *scan backlog* increases every time the system reads data from the PCMCIA card it means that the system is not transferring data fast enough. It is therefore necessary to either increase the buffer size, or change the number of scans per cycle, or decrease the sampling rate. Two *IF* structures regulate the number of scans to be retrieved from the data buffer and the display of the number of samples that is shown by a digital indicator on the front panel. The first one shown in FIGURE 7 for the *FALSE* case, since this is the case that is always utilized by the program. This *IF* structure is redundant, since the number of samples to be used to timeout the acquisition procedure is fixed to 300,000. However, there is a possibility that the user may want to replace the numeric constant with a digital control. In that case the *IF* structure would be fully utilized since the input would initially be set to 0 and then modified by the user to the desired duration of the data collection. The output of this structure goes to the *NUMBER OF SCANS* input of the *AI READ* module and adjusts the number of scans according to possible scan backlogs. The second *IF* structure passes an output to the digital indicator that shows the number of samples read at a certain point in time. This number is obtained by dividing the *MARK AFTER WRITE* output of the *WRITE FILE* module by the number of bytes per scan (the number of channels multiplied by the number of bytes per datum).

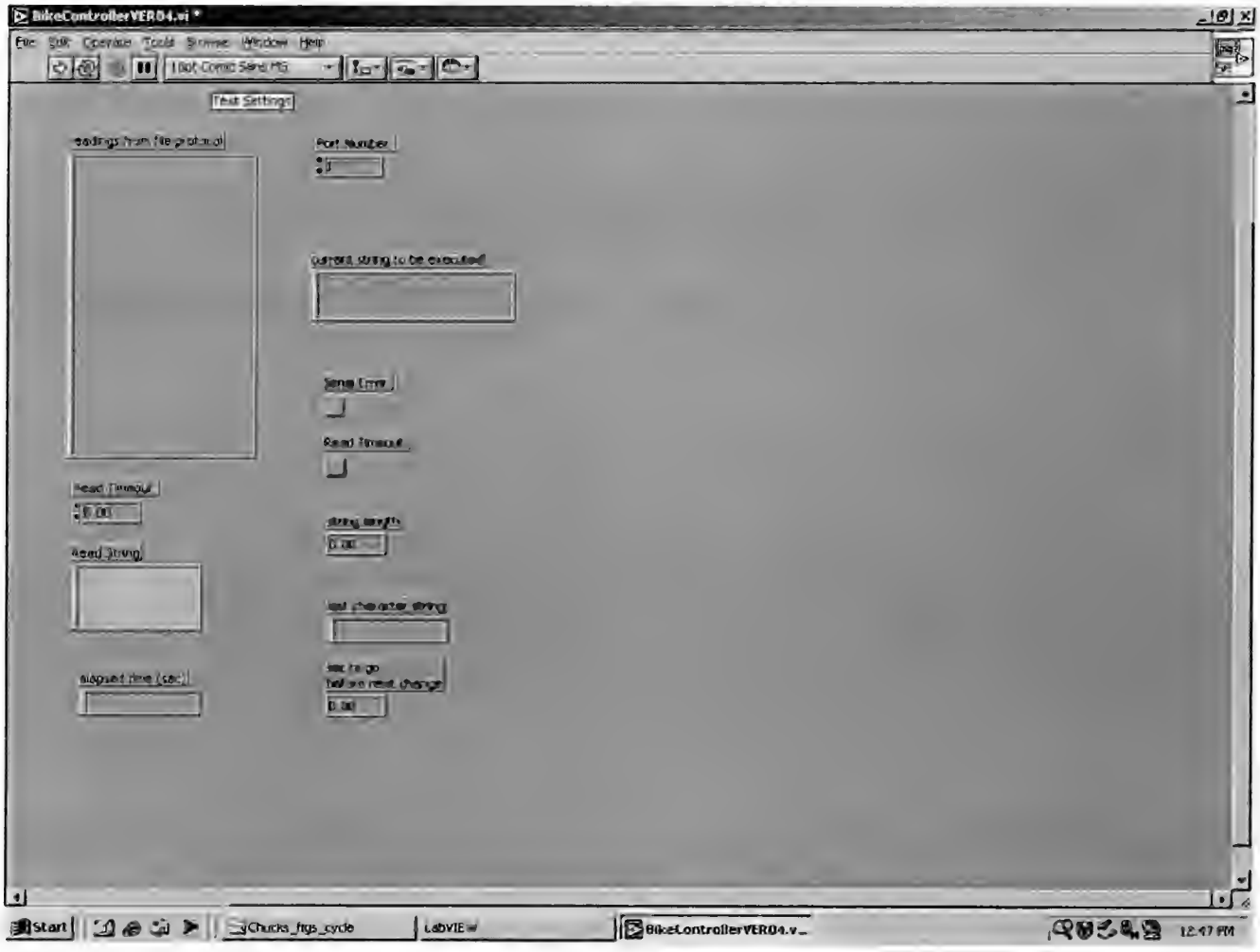
The VI diagram is completed by three modules devoted to 1) closing the data file, 2) de-allocating the resources utilized to handle the acquisition, and 3) managing possible errors. The *GENERAL ERROR HANDLER* module is not linked at this time to

a string indicator in the front panel, since there were no problems during the testing phases of this software. However, it is left in place in case the user wants to test these VIs under more challenging conditions (e.g., higher sampling rate or overload of the system as a consequence of other modules that are run simultaneously). In that case an indicator can be easily added to the front panel and linked to the *MESSAGE* output of the *GENERAL ERROR HANDLER* module.

BikeController a VI to Drive the CardiO2 Cycle

The last of the VIs designed and implemented by the NMRC for the USARIEM is *BikeController.* This VI provides a way to control CardiO2 Cycle and switch among modes of exercise at any point in time. FIGURE 8 shows the front panel of this VI with the controls and indicators designed to monitor the correct functioning of CardiO2 Cycle when driven by sequences of commands sent via the serial port.

FIGURE 8: Front panel of BikeController allows the user to drive the CardiO2 Cycle according to the desired experimental protocol.

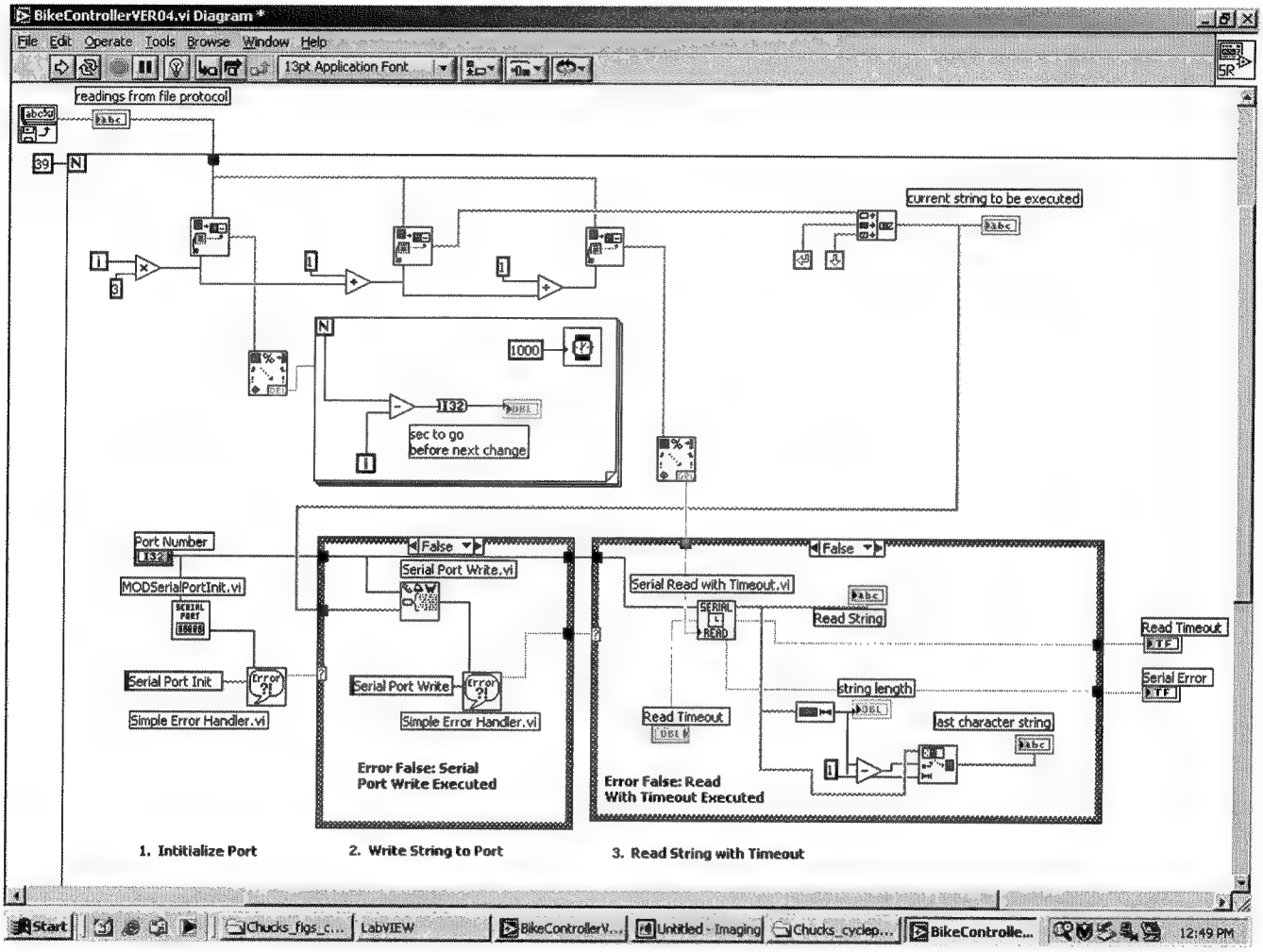


The front panel of this VI includes two controls and several indicators. The controls allow setting the serial port number (1 in the current configuration) and the timeout utilized when reading the responses from CardiO2 Cycle to the serial port command. The indicators include four string indicators that are devoted to: 1) showing the content of the protocol, 2) indicating the last command sent to CardiO2 Cycle, 3) showing the response received from CardiO2 Cycle, and 4) controlling the last character of the string received from CardiO2 Cycle (this last indicator is useful to troubleshoot transmission errors). The indicators also include three digital outputs that are used 1) to show to the user the time elapsed from the beginning of the experiment, 2) to provide a count down to the next change of exercise mode, and 3) to indicate the

length of the string read via the serial port (also this indicator is used to test for possible transmission errors). Finally, two square buttons are used to indicate the occurrence of a timeout error or a serial read error.

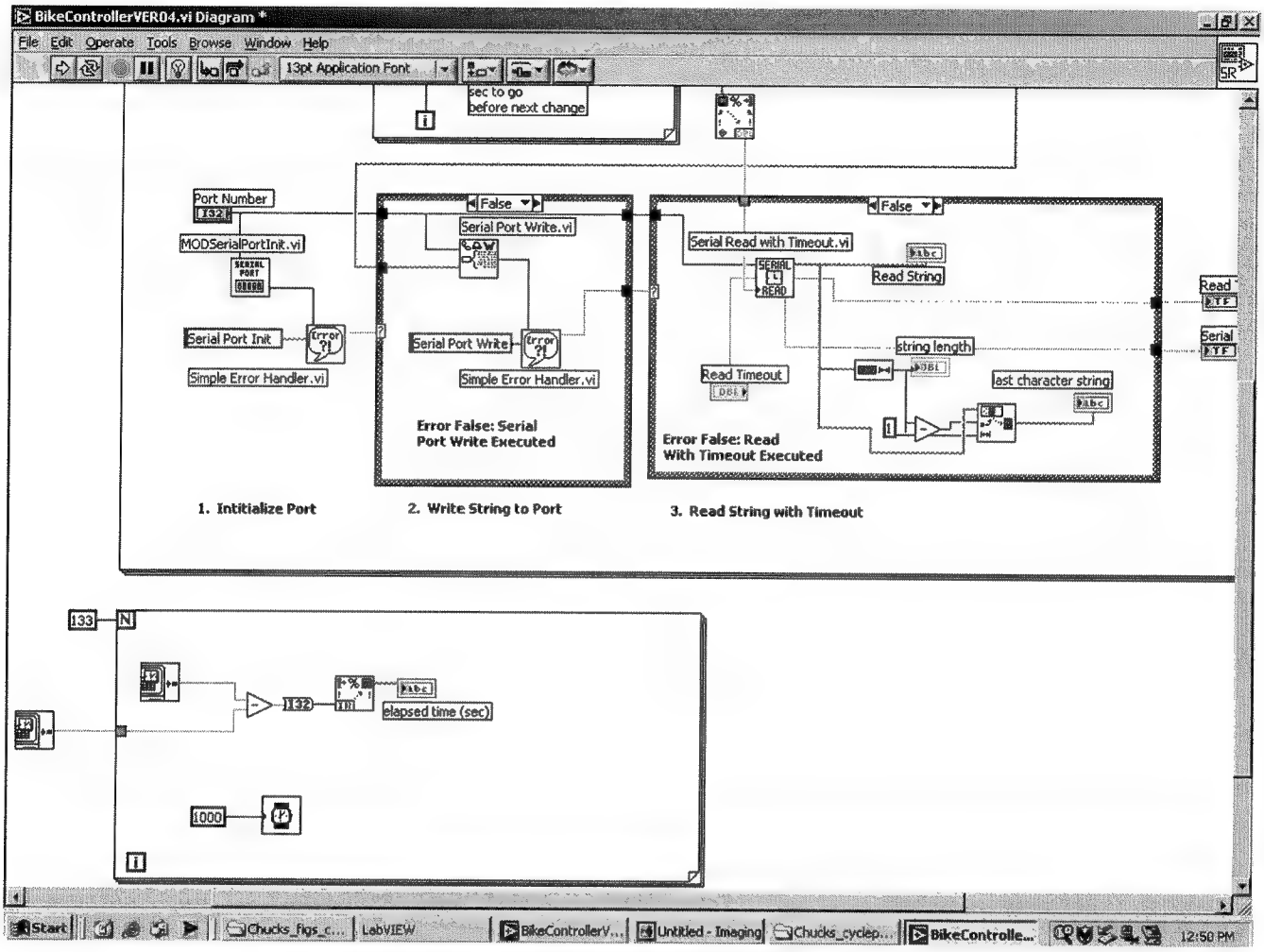
The VI diagram is shown in FIGURE 9. When this VI is run, the *READ LINES FROMFILE* module opens a dialog box to choose the protocol to be executed and reads the lines of the script file generated by *ScriptGen.* Three *PICK LINE AND APPEND* modules are used to select three consecutive lines of the script file. These lines contain time constant commands to be sent to CardiO2 Cycle, and a number of bytes to be returned. The time constant is used as input of a *FOR LOOP* that handles the countdown indicator and changes the number shown by the corresponding digital indicator every second. The command is passed to an *IF* structure that contains a *SERIAL* PORT *WRITE* module. The execution of this structure is preceded by that of a *SERIAL* PORT *INIT* module that initializes the serial port to the modality required by CardiO2 Cycle. A second *IF* structure performs the operations related to reading the response from CardiO2 Cycle to the command just sent. A *SERIAL PORT READ* module is used and three indicators are set according to the outputs of this module. The execution of the *IF* structures as shown in FIGURE 9 is conditional to the absence of error conditions. Also, the output of the second structure consists on two indicators related to timeout and generic error conditions. It is worth emphasizing that the user can separately check the writing and reading operations by double-clicking on the *SERIAL PORT WRITE* and *SERIAL PORT READ* modules. A front panel shows up for each module that allows the user to input single commands using the keyboard.

FIGURE 9: Diagram of BikeController. This part of the diagram is devoted to handle operations related to sending commands to the CardiO2 Cycle and then reading the responses via the serial port.



Finally, a separate part of the VI diagram (see FIGURE 10) is devoted to display the time elapsed from the beginning of the experiment. The *FOR LOOP* shown in Figure 10 runs independently on the first part of the VI diagram. The time elapsed from the beginning of the experiment is computed using the *GET DATE/TIME IN SECONDS* module that is run right at the time the VI is executed and every second until the *FOR LOOP* is fully executed.

FIGURE 10: Diagram of BikeController. This part of the diagram is devoted to display of the time elapsed from the beginning of the experiment.



APPENDIX 1: SOFTWARE COMMANDS USED BY CARDIO2 CYCLE

$BDA	Begin Data Acquisition - starts acquiring data for minimum, and accumulated work values. Note that this command does not reset any of the registers.
$CAL (100-1000) (200-1500)	CALibrate is used to calibrate torque transducer. After $NULlmg the CardiO2 Cycle ergometer, the flywheel is locked (power is still disabled). Then a calibration torque of 100 to 1000 inch pounds is applied to the pedal crank. The command "$CAL n" is issued, where "n" is the applied torque. A number representing torque in internal units is returned for diagnostic purposes.
$DAT (value)	DATA parameter - loads the EEPROM data buffer with a 16 bit value (0 to 65535) for saving in an EEPROM register. See the $SVM command for writing value.
$EDA	End Data Acquisition - accumulated work values. The values will stop being updated when this command has been issued.
$GTM (reg #)	Get Memory register - returns the value of the data word located the selected EEPROM register. Valid register values are 0 to
$NUL	NUL 1 sets null point of torque transducer. Command must be executed prior to every exercise session. Do not apply torque to pedals and disable power section when the command is issued. Command returns a number in the range 500-2500 for diagnostic purposes.
$RCL	ReadCaL reads the current calibration gain value (which may be different from the value saved in the EEPRO).
$RDA	Reset Data Acquisition - resets the register for minimum, peak, accumulated work. If data acquisition is currently underway, it continues immediately after the registers have been reset.
$RLT	Read Lower Target.
$RMD	ReadMoDe returns Work rate, Seed, Torque, or WIN ate.
$RMS	Read Minimum Speed.
$RNL	Read NuLl reads current null value.
$RPW	Read Polder stage reports ON/OFF status.
$RSG	Read Strain Gauge.
$RSP	Read actual Speed.
$RSS	Read Set Speed.
$RTC	Read Time Constant.
$RTQ	Read actual Torque.
$RTS	Read Torque Setting.
$RUN W,S,T,G	RUN enables the CardiO2 Cycle ergometer power section.
$RUT	Read Upper Target.

$RWA	Read Work Accumulated - returns total amount of work done (in
	watts-seconds or joules) since data acquisition was started. This
	command does not halt data acquisition if it is occurring. Note
	the maximum number of joules that can be returned is approx
	65000, so if a long duration test is planned, the host should query periodically and if approaching the maximum should save the
	current value for later summing and issue a $RDA command to reset the register.
$RWK	Read actual Work rate.
$RWM	Read Work Minimum - returns the minimum value of work since
	data acquisition was started. This command does not interrupt
	acquisition. If no data have been gathered prior to issuing this
	command, it will return a value of 65535.
$RWP	Read Work Peak - returns the peak value of work since data
	acquisition was started. This command does not interrupt
	acquisition.
$RWS	Read work Rate Setting.
$SLT (0-125)	SetLower Target sets lower speed boundary for "too slow" LED
	(turtle). The CardiO2 Cycle ergometer picks setting when zero is
	set.
$SMD (W,S,T,G)	SetMoDe sets mode of regulation to constant Work, Constant Speed, constant Torque, or WINgate
$SMS (0-125)	SetMinSpeed sets minimum speed the CardiO2 Cycle ergometer runs when subject exercises
	below regulation point. Zero is a special setting that tells the cycle
	ergometer to pick "optimized" setting for current exercise set point
	This is also the default. Units:RPM
$SSP 0-125	SetSPeed sets exercise seed in ISOkinetic mode. Units: RPM.
$STC (0-7)	SetTime Constant sets time constant for flywheel effect. With lower setting the Cardio2 Cycle ergometer responds
	to set point changes more rapidly. Generally, use low time
	constants with low exercise set points and high time
	constants with high exercise set points. Zero, a special setting that tells the Cardio2 Cycle ergometer to pick a

	setting for current exercise set point, is the default.
$STP	STOP disables the Cardio2 Cycle ergometer power section.
$STQ (0-1000)	SetTorQue sets exercise resistance torque in TORQUE mode. Units: inch- pounds.
$SUT (0-125)	SetUpperTarget sets upper speed boundary for "too fast" LED
	(rabbit). The Cardio2 Cycle ergometer picks setting when zero is
	set.
$SVC	SaVeCal saves calibration information in non-volatile memory for use in future sessions
$SVM (reg #)	Save Memory register - writes the value currently loaded into the
	EEPROM data buffer to the selected EEPROM register. Valid
	register values are 0 to 127. Register 1 holds the system calibration
	value, 2 holds the null value, and 127 contains an EEPROM
	checksum used to insure data integrity. All other registers are
	available for general purpose non-volatile storage. When writing to
	any EEPROM register the user is responsible for correctly updating
	the checksum. This may be done automatically by issuing a $SVN
	command to save the null value, or by summing (module 16) the
	values in registers 0-126, and writing the sum to register 127. If this
	is not done correctly, the system will generate a checksum error
	message.
$SVN	SaVe Null saves null information in non-volatile memory for use in
	future sessions.
$SWK (0-500)	SetWorK sets exericse work rate in work mode. Units: watts. The
	range can be set to 0-1000 for the Win ate software only.
$VER	VERsion returns software version number.
$WIN (50-1000)	WINgate sets mode to wingate, starts ergometer (if not already
	running) and initializes data gathering. The argument is the torque
	level in inch-pounds. If omitted, the last (isotonic or wingate)
	torque setting will be used. This command is equivalent to issuing
	the following commands simultaneously: $SMD G, $STQ (torque),
	$RDA, $BDA, $RUN.

APPENDIX II: TECHNICAL DRAWINGS OF HARDWARE MODIFICATIONS

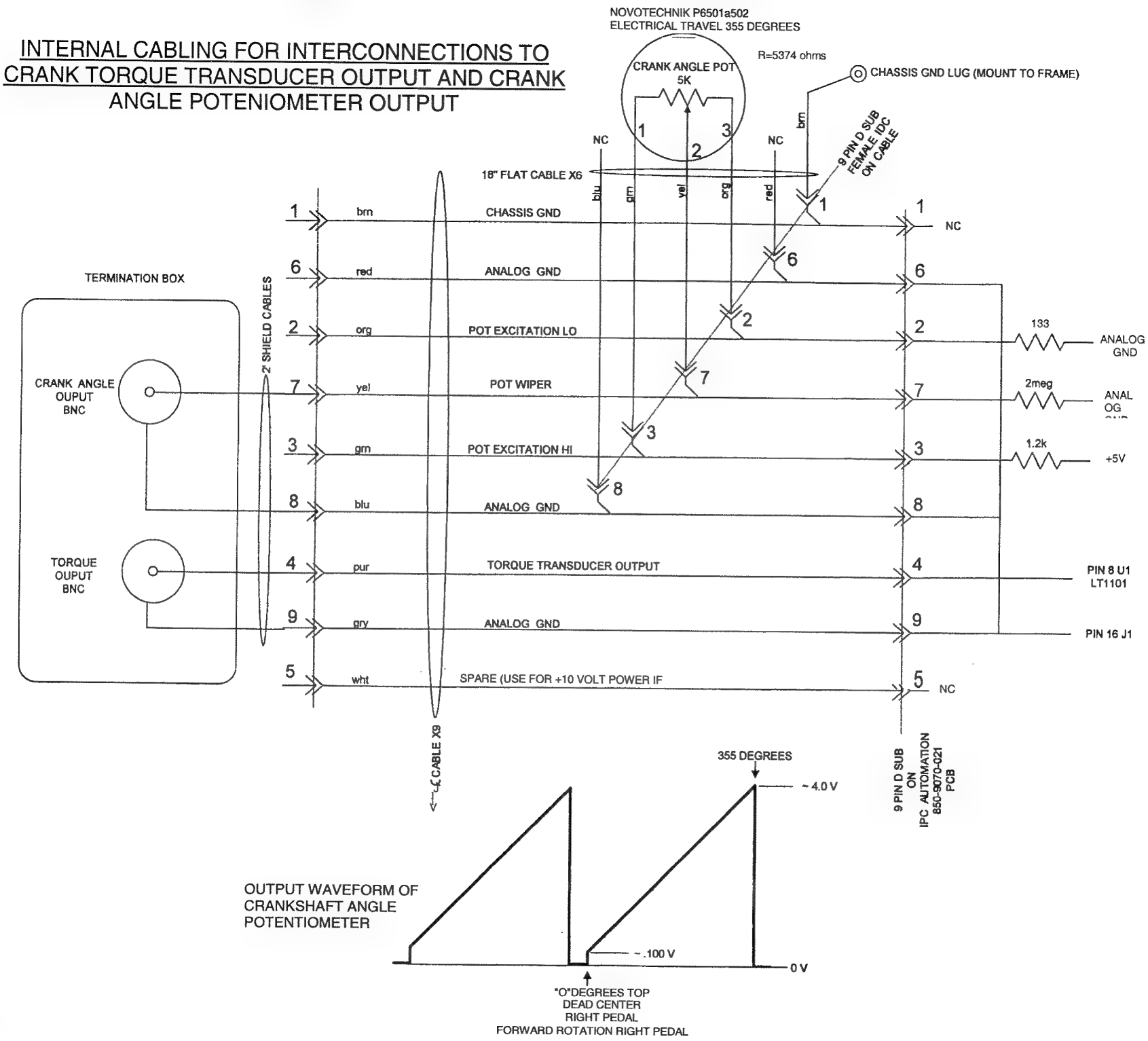
1. Internal Cables For Interconnections To Crank Torque Transducer Output And Crank Angle Potentiometer Output

2. Crankshaft Timing Modifications

3. Potentiometer Timing Pulley And Bracket Assembly (View One)

4. Potentiometer Timing Pulley And Bracket Assembly (View Two)

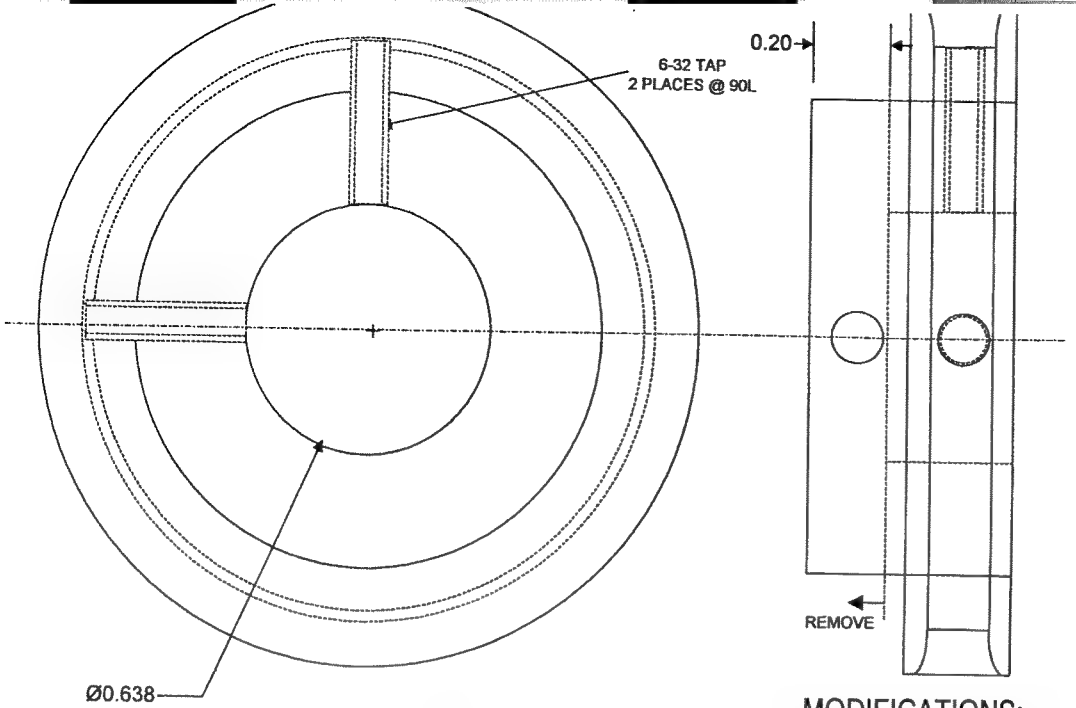
INTERNAL CABLING FOR INTERCONNECTIONS TO CRANK TORQUE TRANSDUCER OUTPUT AND CRANK ANGLE POTENIOMETER OUTPUT



CRANKSHAFT TIMING PULLEY MODIFICATIONS

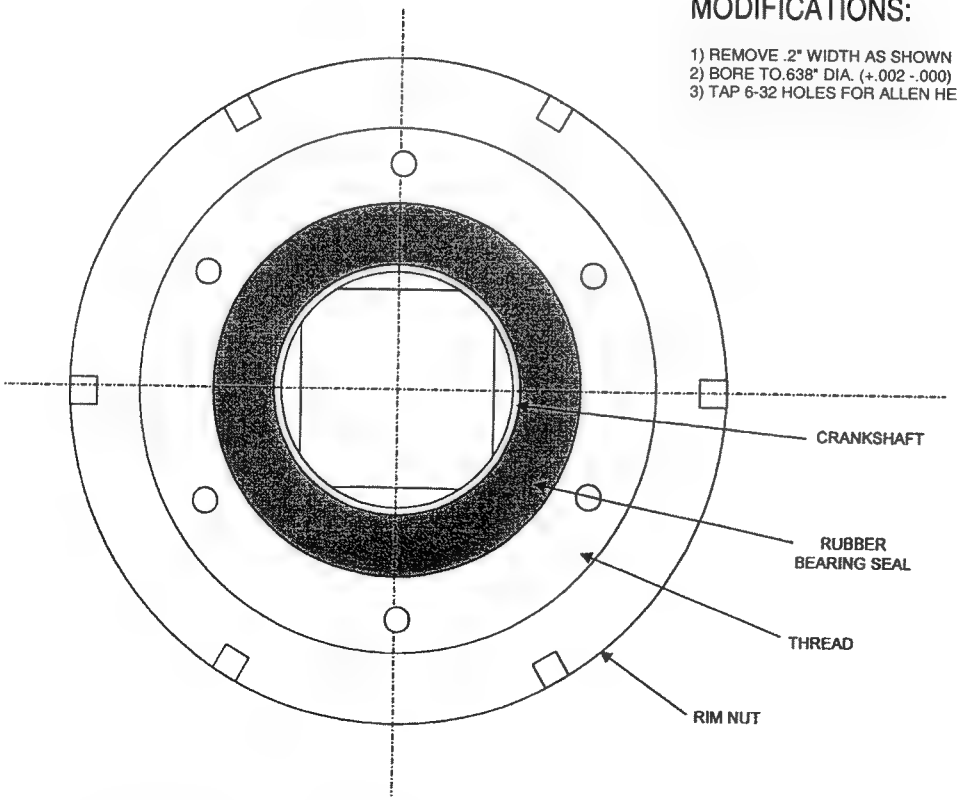
STOCK DRIVE PRODUCTS STERLING
INSTRUMENT TIMING BELT PULLEY.08 PITCH A
6N16-060DF128 sdp page 1-10

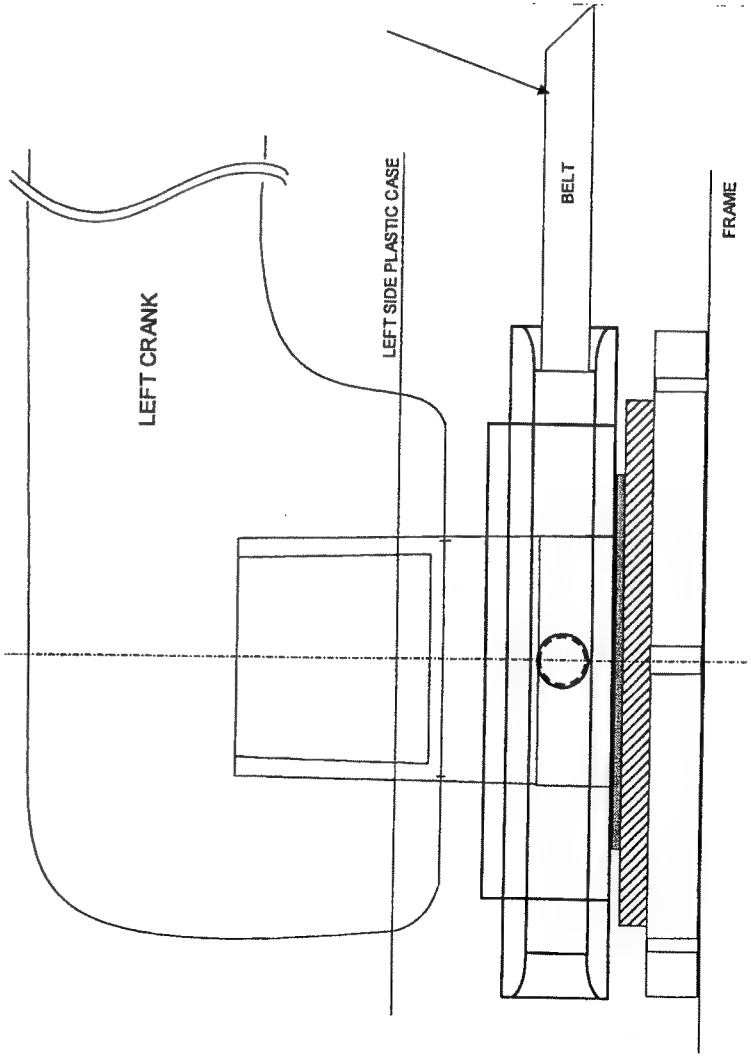
USE WITH URETHANE TIMING BELT: A 6B 16-135-012



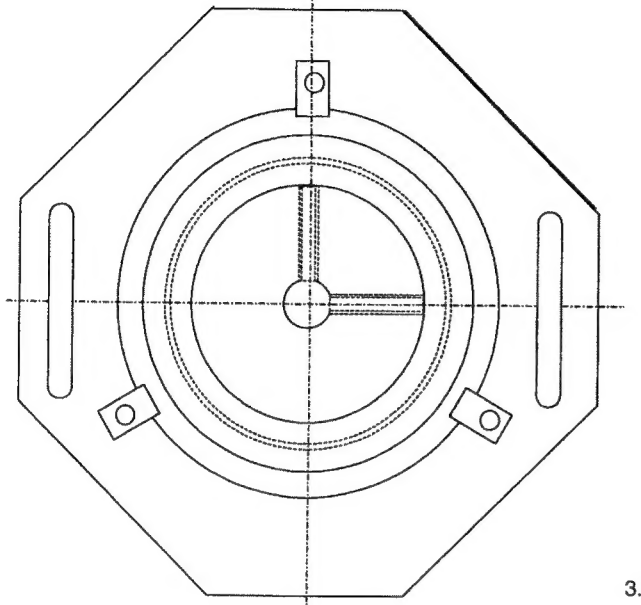
MODIFICATIONS:

1) REMOVE .2" WIDTH AS SHOWN
2) BORE TO.638" DIA. (+.002 -.000)
3) TAP 6-32 HOLES FOR ALLEN HEAD

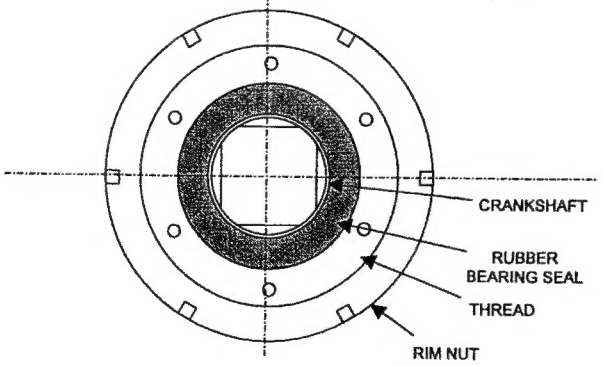




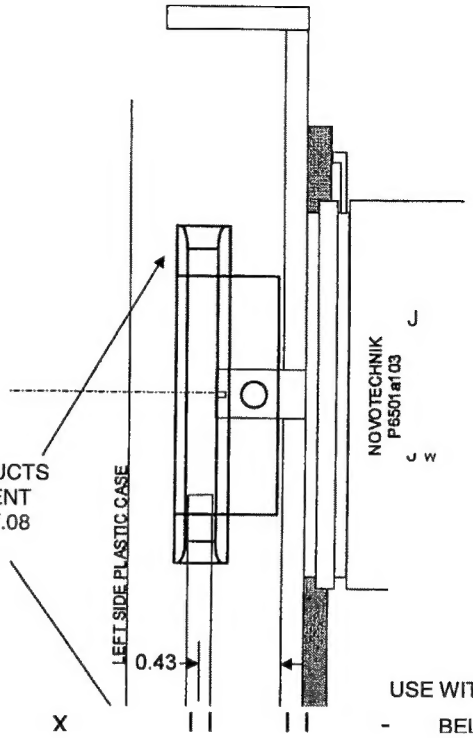
POTENTIOMETER TIMING PULLEY + BRACKET ASSEMBLY



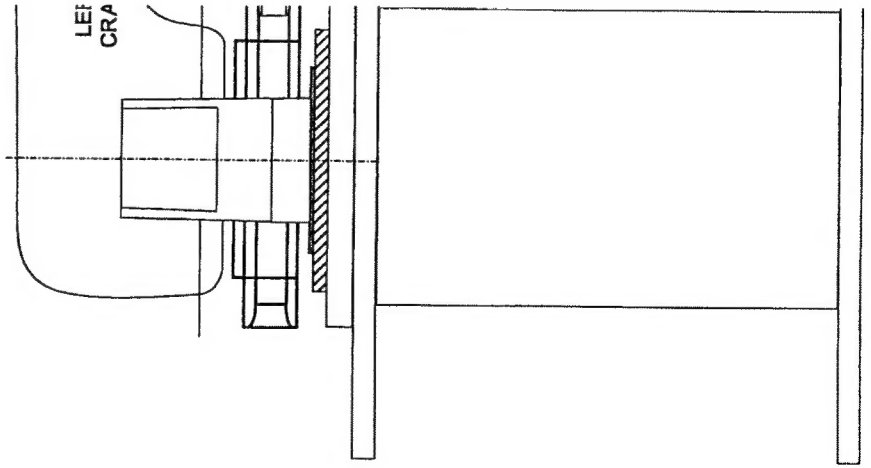
3.00

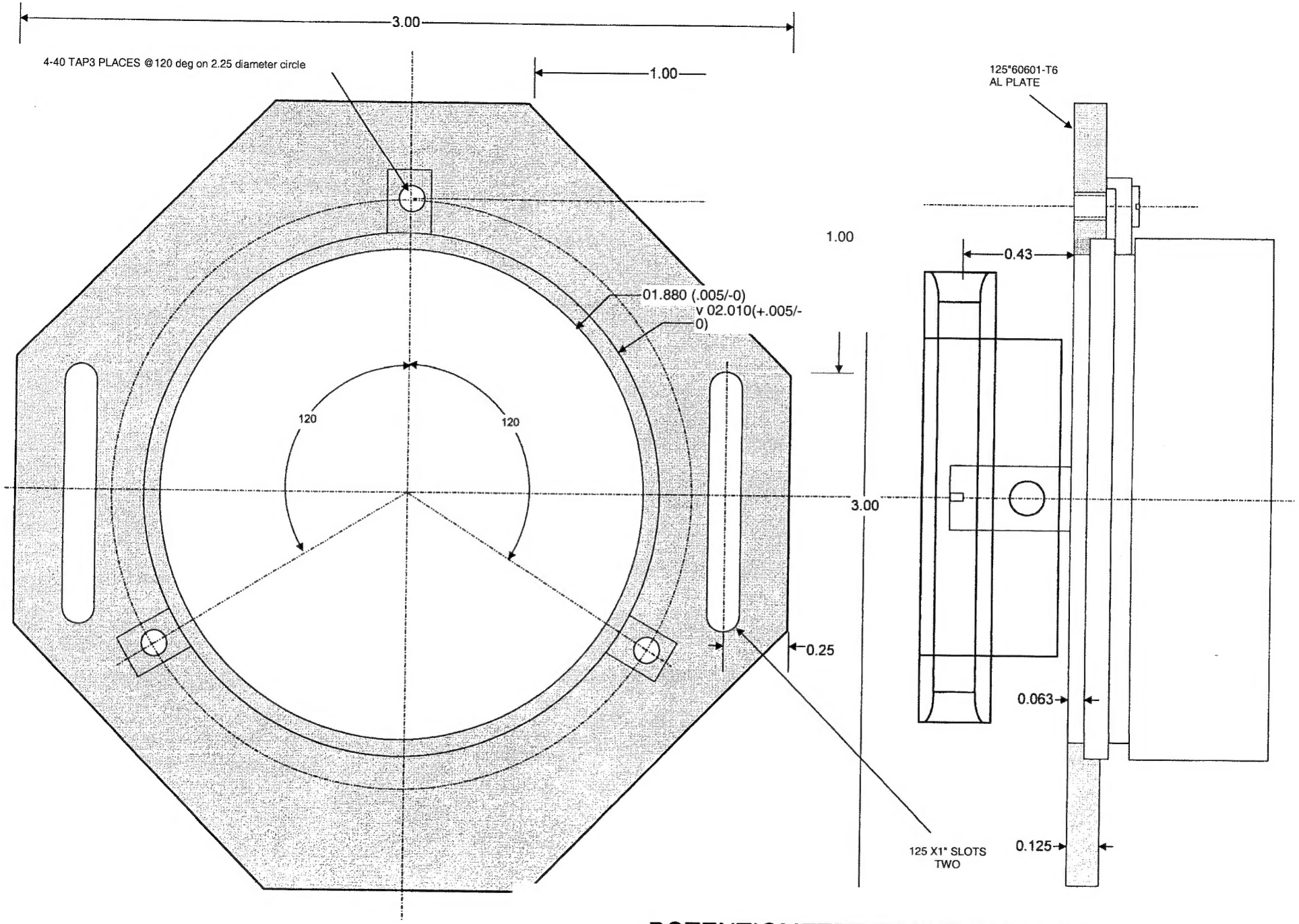


STOCK DRIVE PRODUCTS
STERLING INSTRUMENT
TIMING BELT PULLEY.08
PITCH A 6N16-
060DF128 sdp page 1-
10



USE WITH URETHANE TIMING
BELT: A 6B 16-135-012
sdp pagel-6





POTENTIOMETER TIMING PULLEY + BRACKET

REFERENCES

1. **Bigland-Ritchie B, Rice CL, Garland SJ and Walsh ML**. Task-Dependent Factors in Fatigue of Human Voluntary Contractions. In: Fatigue: Neural and Muscular Mechanisms, edited by Gandevia SC, Enoka RM, McComas AJ, Stuart DG and Thomas CK. New York: Plenum Press, 1995, p. 361-380.

2. **Fulco CS, Lewis SF, Frykman P, Boushel R, Smith S, Harman EA, Cymerman A and Pandolf KB**. Quantitation of progressive muscle fatigue during dynamic leg exercise in humans. *J Appl Physiol* 79: 2154-2162, 1995.

3. **Fulco CS, Lewis SF, Frykman P, Boushel R, Smith S, Harman EA, Cymerman A and Pandolf KB**. Muscle fatigue and exhaustion during dynamic leg exercise in normoxia and hypobaric hypoxia. *J Appl Physiol* 81: 1891-1900, 1996.

4. **Fulco CS, Rock PB, Muza SR, Lammi E, Cymerman A and Lewis SF**. Reproducible voluntary muscle performance during constant work rate dynamic leg exercise. *Int J Sports Med* 21: 1-5, 2000.

5. Lewis, S. F., Almazeedi, S., Bonato, P., Soares, R., Lammi, E., Roy, S. H., and Fulco, C. S. Slower recovery of myoelectric activity and force after lower intensity exhaustive dynamic exercise. FASEB Journal 14, A282. 2000.

6. **Lewis SF and Fulco CS**. A new approach to studying muscle fatigue and factors affecting performance during dynamic exercise in humans. In: Exercise and Sport Sciences Reviews, edited by Holloszy JO. Baltimore,MD: Williams & Wilkins, 1998, p. 91-116.

7. Lewis, S. F., Fulco, C. S., Frykman, P., Boushel, R., Smith, S., Harman, E., Cymerman, A., and Pandolf, K. B. Muscle fatigue does not impact pressor responses during dynamic leg exercise. FASEB Journal 9, a360. 1995.

8. Lewis, S. F., Lammi, E., Soares, R., Lieberman, H. R., Taub, I. A., and Fulco, C. S. Slower muscle fatigue rate with increased carbohydrate availability during prolonged leg exercise. American College of Sports Medicine 30(5), s208. 1998.

9. Lewis, S. F., Soares, R., Lammi, E., Baker-Fulco, C. J., Lieberman, H. R., and Fulco, C. S. Knee extensor muscle endurance and caffeine dose before and after exhaustive dynamic exercise. American College of Sports Medicine 33, S426. 2001.